Defending Adversarial Examples via DNN Bottleneck Reinforcement

Wenqing Liu, Miaojing Shi, Teddy Furon, Li Li

▶ **To cite this version:**

Wenqing Liu, Miaojing Shi, Teddy Furon, Li Li. Defending Adversarial Examples via DNN Bottleneck Reinforcement. ACM Multimedia Conference 2020, Oct 2020, Seattle, United States. pp.1930-1938, 10.1145/3394171.3413825 . hal-02912189

HAL Id: hal-02912189
https://hal.science/hal-02912189

Submitted on 20 Nov 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Defending Adversarial Examples via DNN Bottleneck Reinforcement

Wenqing Liu*
College of Electronic and Information Engineering
Tongji University, Shanghai, China
liuwenqing@tongji.edu.cn

Miaojing Shi
King's College London
London, UK
miaojing.shi@kcl.ac.uk

Teddy Furon
Univ. Rennes, Inria, CNRS, IRISA
Rennes, France
teddy.furon@inria.fr

Li Li†
College of Electronic and Information Engineering
Institute of Intelligent Science and Technology
Tongji University, lili@tongji.edu.cn

ABSTRACT

This paper presents a DNN bottleneck reinforcement scheme to alleviate the vulnerability of Deep Neural Networks (DNN) against adversarial attacks. Typical DNN classifiers encode the input image into a compressed latent representation more suitable for inference. This information bottleneck makes a trade-off between the image-specific structure and class-specific information in an image. By reinforcing the former while maintaining the latter, any redundant information, be it adversarial or not, should be removed from the latent representation. Hence, this paper proposes to jointly train an auto-encoder (AE) sharing the same encoding weights with the visual classifier. In order to reinforce the information bottleneck, we introduce the multi-scale low-pass objective and multi-scale high-frequency communication for better frequency steering in the network. Unlike existing approaches, our scheme is the first reforming defense per se which keeps the classifier structure untouched without appending any pre-processing head and is trained with clean images only. Extensive experiments on MNIST, CIFAR-10 and ImageNet demonstrate the strong defense of our method against various adversarial attacks.

CCS CONCEPTS

• **Security and privacy → Software and application security**;
• **Computing methodologies → Object recognition**.

KEYWORDS

DNN, adversarial attacks, information bottleneck, auto-encoder

ACM Reference Format:
Wenqing Liu, Miaojing Shi, Teddy Furon, and Li Li. 2020. Defending Adversarial Examples via DNN Bottleneck Reinforcement. In *Proceedings of the 28th ACM International Conference on Multimedia (MM '20), October 12–16, 2020, Seattle, WA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3394171.3413825

*Wenqing Liu and Miaojing Shi contributed equally.
†Corresponding author.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
MM '20, October 12–16, 2020, Seattle, WA, USA
© 2020 Association for Computing Machinery.
ACM ISBN 978-1-4503-7988-5/20/10…$15.00
https://doi.org/10.1145/3394171.3413825

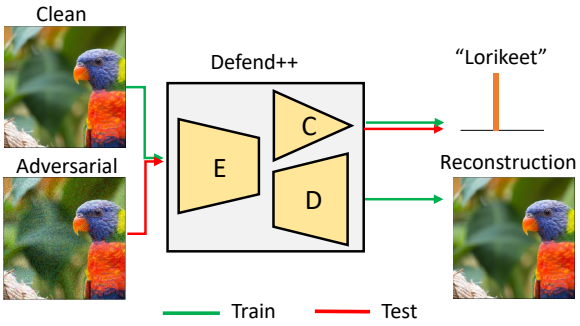

Figure 1: Illustration of our Defend++: defending adversarial examples via DNN bottleneck reinforcement. The structure of a classifier is untouched but split into two parts (E+C). A specifically designed D-ecoder is appended during the training to form an auto-encoder (E+D).The structure is trained on clean images only to jointly minimize the classification (E+C) and the reconstruction (E+D) losses.

1 INTRODUCTION

Deep learning achieves excellent performance in various multimedia and computer vision tasks [12], but it has weaknesses. Researchers have observed that specially crafted perturbations to images/videos lead to total failure of visual recognition [6, 11, 28]. Surprisingly, these perturbations have very small amplitude so they are barely perceptible. It means that in the image space, natural images lie close to the class boundary and are easy to be perturbed.

The literature on adversarial images considers two scenarios. In the while-box scenario, the attacker knows everything about the classifier. The backpropagation computes the gradient of this classifier's loss w.r.t. the image pixel values [31]. It indicates the infinitesimal perturbation in the image space dragging the image closer to the class boundary [28]. This operation can be iterated until the image crosses the boundary so that the model prediction is wrong [13]. Adversarial examples also happen to transfer across different models. In the black-box scenario, the attacker is oblivious to the target model internals. Yet, by observing the target's outputs for many inputs, the attacker can train his own model mimicking

the target. By leveraging transferability, a white-box attack on his own classifier often deludes other black-box models [21].

Many defenses have been proposed but they resort to few different mechanisms. One idea coined as *gradient masking* [2, 23] targets the core of white-box attacks, the computation of gradient, by blocking the backpropagations. Yet, gradient masking has been shown to give a false sense of security [4]: this defense is not efficient against black-box attacks, which are free from any gradient computation on the target model.

Adversarial training robustifies the classifier by pushing the class boundary away so that adversarial images lie in the true class region [6, 28]. The fine-tuning of model parameters is time-consuming. This is why adversarial training usually considers fast but simple attacks. Its strength against more elaborated attacks remains questionable.

A good defense should not be dedicated to any specific attack or mechanism, which we conceptualize as its *universality*. The class boundaries learned by the classifier are valid over the manifold of natural images, while perturbations look like noise and perturbed images are not lying on the manifold of natural images; the inference of the classifier can be awfully wrong when probing images off this manifold. This motivates pre-processing the input image in front of the classifier [21]. The goal is to reform suspicious images, i.e. to project them back onto the manifold of natural images [1]. The most popular choice for this front-end pre-processing is an auto-encoder (AE) [1, 7, 15]. It is usually trained with adversarial examples forged by many attacks. Few-shot learning can be adopted to accelerate this process [18]. Nevertheless, training it with clean images can strongly enforce the universality of the defense. This is recently achieved in [10].

This paper proposes to defend adversarial examples via DNN bottleneck reinforcement, which we call it Defend++. It pertains to the defense trend of universality yet is not a separated front-end input reformer. Instead, our idea is to merge the reformer (i.e. AE) with the early layers of the classifier so that both functionalities are jointly trained. Once training is finished, the inference of the classifier is disjointed from the AE (see Fig. 1). Our architecture can be seen in two ways: it is a classifier working directly over the internal latent space representation of the AE (instead of being connected at its output). From another point of view, it is a visual classifier whose first layers are jointly trained for both classification and reconstruction.

Our motivation is based on the following view of a classifier. A DNN encodes an image into a compressed latent representation compressing the image-specific structure (i.e. edges, corners, and regions) to concentrate the class-specific information (i.e. object size, location, and common traits of the class). It has been theoretically justified as necessary for better classification by the concept of information bottleneck [26]. If we could reinforce the image-specific structure while maintaining the class-specific information, any redundant information, adversary or not, would be removed without harming the prediction. This information bottleneck reinforcement has to take place inside the network.

Due to the fact that adversarial perturbations happen to be high frequency in nature, we reinforce the information bottleneck with frequency steering in the AE. We introduce the 1) multi-scale low-pass objective to gradually reconstruct the image from low-level structure to high-level details; 2) multi-scale high-frequency communication to separate the prediction of high-frequency bands from lowpass images and connect them to different parts of the encoder for high-level communications. To achieve the universality, the training of Defend++ deals with clean images only. We do not alter the original classifier structure and no pre-processing is placed up front. Details of the classifier and training strategy are public. Our method is both attack agnostic and scenario agnostic.

The contribution of Defend++ is three-fold:

- We improve the universal defensive ability of the classifier by jointly learning an AE with it sharing the same encoding weights and on clean images only.
- We reinforce the network information bottleneck by introducing the multi-scale low-pass objective and multi-scale high-frequency communication for the AE.
- We conduct extensive experiments on MNIST, CIFAR-10 and ImageNet benchmarks and demonstrate strong defense of our Defend++ against various adversarial attacks.

To our knowledge, Defend++ is the first reforming defense that requires neither modification on the architecture of classification net, nor adversarial data for training.

2 RELATED WORKS

This section surveys adversarial learning from the attack and defense perspective, respectively. We mainly review white-box attacks as they are the hardest to defend.

2.1 Attack methods

All white-box attacks benefit from the cheap computation of the gradient direction of the loss thanks to the backpropagation. In the initial discovery of adversarial examples in [28], the Fast Gradient Sign Method (FGSM) minimizes a first-order approximation of the loss over the ball of ℓ_∞ norm ϵ. Kurakin et al. [13] further introduce a basic iterative method (BIM) by repeating FGSM for several steps. Carlini-Wagner [4] looks for the adversarial perturbation of minimal ℓ_2 norm. The solution is given by a Lagrangian formulation that combines the gradients of the loss and the Euclidean distortion. Yet, C&W requires a high number of iterations as several Lagrangian multiplier values are tested. Recently, Rony et al. [25] introduce an efficient attack quickly finding an adversarial image and then refining it to minimize its distortion. This is done by decoupling the direction and norm (DDN) of the perturbations.

2.2 Defense methods

The first counter-attack, *gradient masking*, blocks the backpropagation, which is at the core of any white-box attack, by introducing some randomization [29], smooth labels [2, 23], soft feature selections [9], and regularizers [22] to make the model output less sensitive to the perturbation on input. The classification structure is often modified in these works [9, 29]. Also, there exists a vulnerability against non gradient-based attacks (black-box) where the attacker is free from differentiating proxies of these methods to circumvent the defense. This is outlined in [4].

Adversarial training is a promising idea where a network is retrained with adversarial images. It performs well on small datasets such as MNIST and CIFAR, but decreases accuracy on large-scale

datasets like ImageNet [13]. It is much costly. An attack has to be mounted against a model, anytime that model is updated, new adversarial images must be generated. This is the reason why adversarial training only uses fast attacks like FGSM which are not very powerful. The status may change in the future with the invention of fast and efficient attacks like DDN [25].

A third idea is to detect adversarial images [14, 17]. This amounts to add the extra class "suspicious images" as a possible output of the classifier. The attack is made harder as it must delude both the detector and the classifier. Training usually needs adversarial examples to minimize the false negative detection rate. This questions their ability to detect new attacks [3]. As far as we know, [19] is the only detector trained on clean images only.

A fourth idea is to utilize a pre-processing head in charge of filtering out any perturbations if existed. This is usually called a reformer. Dziugaite et al. [5] report that JPEG compression can reverse small adversarial perturbations. Several variants, such as feature squeezing [30], pixel defend [27] and distillation [16], are further proposed to mitigate adversarial attacks. Meng et al. [21] see the reformer as a projection of the input onto the manifold of clean data. Gu et al. [7] were the first to propose an AE as a reformer. Liao et al. [15] introduce a high-level representation guided denoiser within the AE. Yet, these advanced reformers mostly need training with adversarial examples, and the same question as for adversarial training and detection arises: how do they perform against attacks not encompassed in their training?

The recent paper [10] shuts down the concern by proposing an efficient reformer (AE) that is trained without adversarial images. Its reformer is still separated from the classifier, and is carefully designed to compress over image bits with additional Gaussian noises. Our AE instead shares the encoding part with the classification network and is jointly trained with the classifier without any additional noises. Moreover, [10] assumes that the attacker knows the classifier but not the reformer. Ours is however totally open to attacker. We believe a complete white-box scenario makes more sense as nothing prevents the attacker to reproduce the reformer.

3 METHOD

3.1 Problem definition

Given a visual recognition task, i.e. classification, the target of this work is to learn a robust model that generalizes on a wide range of clean images and defenses against various adversarial examples. In our setting, the model defensive ability can only be improved during classification training without whistles and bells. This is very challenging, but any effective method derived from this setting can be especially helpful.

3.2 Motivation

A typical classification network consists of multiple convolutional layers and several fully connected (fc) layers. The series of convolutional layers is regarded as an encoder producing a latent representation of the image, while fc layers act as a classifier making prediction upon it. This partition is very schematic and indeed any split of the network in two can be loosely seen as an encoder followed by a classifier.

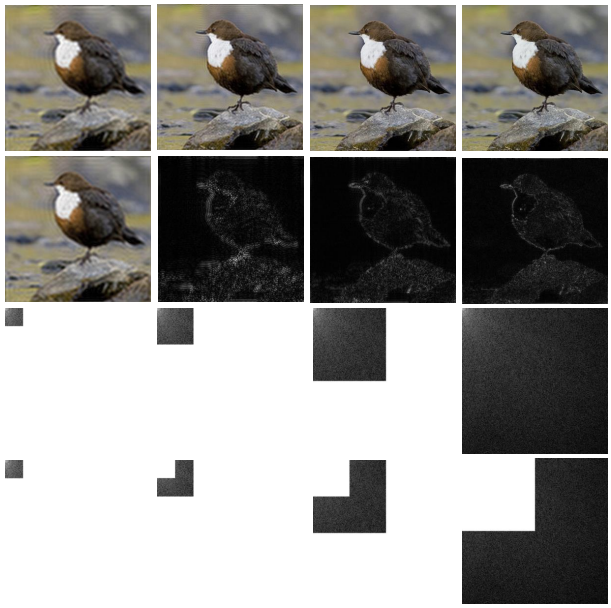

Figure 2: First and third row: lowpass filtered images and corresponding frequency maps via DCT. Second and fourth row: Pyramid images and corresponding frequency maps via DCT.

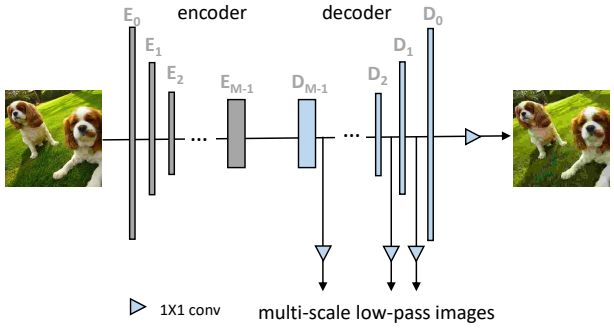

Figure 3: A simple auto-encoder (AE) with hourglass structure. We use E and D to denote the encoding and decoding block, respectively: E_{M-1} (D_{M-1}) signifies a stack of encoding (decoding) blocks, while the others are with single block.

This latent representation contains two-fold information: the image structural information, i.e. edges, corners and regions, that are most visually distinguishable; the class-specific information, i.e. object size, location and appearance, that are most similar to others of the same class. There exist competitions between the two forces due to the nature of classification training. Intuitively, if we could reinforce the first fold while retaining the second fold information, any redundant information should be removed from the image representation and does no harm to the network prediction.

Referring to the adversary generation process, the imperceptible perturbation can be regarded as additional noise with some particular structures. It does not affect the original structure of the image but rather reflects the redundant information of the image [10]. A common approach to remove image redundancy is to

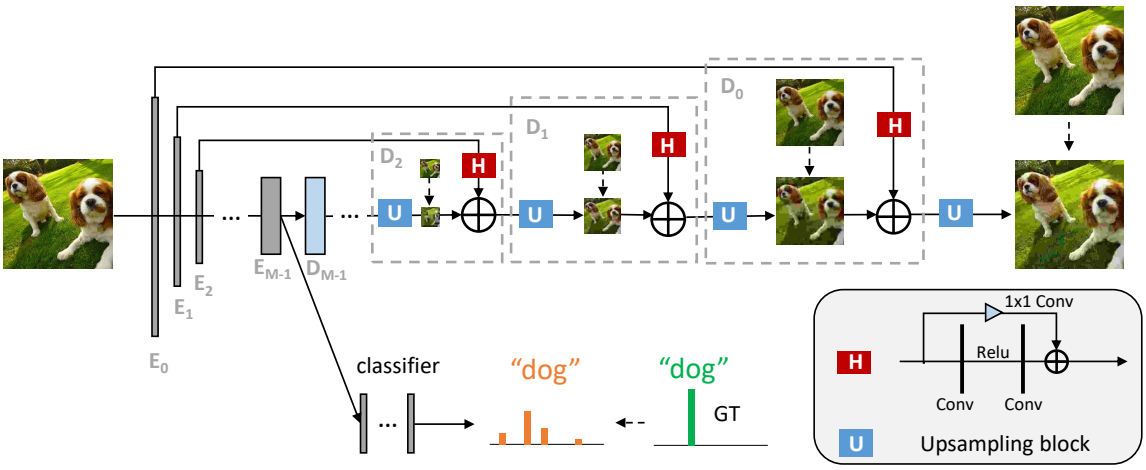

Figure 4: Overview of the reconstruction part of Defend++. The original image is low-pass filtered via DCT to create multi-scale ground truth for the decoder learning. Each decoding block D is formed with U and H modules just like in a Laplace pyramid to separately tune the importance of each frequency band.

train an auto-encoder (AE). Previous methods employ AE for adversarial defense [7, 10, 15] as an independent network for input transformation. In our Defend++, the AE is jointly learnt with the classification net on clean images only. Another difference is that we do not modify the architecture of the classifier once it is given. The encoder of the AE is indeed composed by the first layers of the classification net. In other words, the weights of these layers are learnt by merging both classification and reconstruction losses.

3.3 DNN bottleneck reinforcement

AEs are neural networks that aim to copy their inputs to their outputs. They work by compressing the input into a latent-space representation, then reconstructing the output from this representation. The former part is conceptualized as an encoder removing the redundant information from the image while the latter part is a decoder reconstructing the image. The encoder consists of a set of convolutional layers downsampling the image while the decoder is a set of deconvolutional layers upsampling the image.

In order to distill the encoded knowledge between the classification net and the AE, we share the encoder of the AE with that of the classification net. Notice that when the classification net is too deep or too shallow, we can cut or extend the encoding blocks of the network considered as the encoder to an appropriate length. The decoder design is an open problem: a straightforward way is to employ the so-called hourglass structure [24] where each decoding block is a mirrored version of the encoding block chaining the layers in reverse order (see Fig. 3). Here we use encoding/decoding blocks to refer to a stack of layers in the network where the input is only downsampled/upsampled once with a factor of 2 in this block. For instance, given a classifier ResNet-18, its encoding block normally consists of two/three residual blocks where the stride of the first convolutional layer in the block is set to 2 to downsample the image; a naive decoding block (e.g. in hourglass structure) reveres the encoding block and replaces its last convolutional layer with a deconvolutional layer of stride 2 to upsample the image.

Hourglass is a very simple AE. Below we first introduce a multi-scale lowpass objective to specifically improve its compressing ability against adversarial/noisy perturbations.The encoder in Fig. 3 is only connected to the decoder through the network bottleneck, which is weak. While the encoder is inherited from the classification net and is untouched in our setting, we are free to design the decoder. In order to better steer the image frequency in encoding blocks, we further reform the decoding blocks into a pyramid structure which separates the high-frequency band from each decoding output and connects it to an encoding block for direct communication.

Multi-scale low-pass objective. Image noises including adversarial perturbations happen to be high frequency in nature. Hence, we believe each frequency band deserves a proper treatment. Inspired by papers [5, 30] filtering out adversarial perturbations in certain extent thanks to the Discrete Cosine Transform (DCT), we propose a multi-scale low-pass objective to reconstruct the image gradually from low-level structure to the high-level details.

Formally, let F represent the frequency map of an image I via the full DCT transform, $F = \text{DCT}(I)$. F has the same size $W \times H$ with I. A lowpass filtered version is generated by removing high frequency coefficients (see Fig. 2 first and third rows). The origin starts from the top-left corner of the map where the DC component lies in. Suppose that the decomposition has M levels with a scale of 2. For the m-th level, $m \in \{0, 1, \ldots, M-1\}$, the frequency map F^m has size $W_m = \frac{W}{2^m}$ and $H_m = \frac{H}{2^m}$ and coefficients $F^m_{wh} = F_{wh}, \forall (w, h) \in \{1, W_m\} \times \{1, H_m\}$. The lowpass filtered images $I^0, I^1, ..., I^{M-1}$ are obtained by applying inverse DCT on the M lowpass frequency maps. Note that I^0 is indeed the original image I.

Compared to using linear interpolation to generate multi-scale ground truth, the proposed way provides a better feature steering to reconstruct different levels of details.

Multi-scale high-frequency communication. Following the notations above, we use $\tilde{I}^0$,..., $\tilde{I}^{M-1}$ to denote the reconstructed output corresponding to the multi-scale lowpass ground truth I^0,..., I^{M-1}. In the hourglass structure, the decoding blocks D_{M-1},..., D_0 are chained in a reverse order of the encoding blocks E_0,..., E_{M-1}

with deconvolutional layers. Notice D_{M-1} denotes a stack of multiple decoding blocks to decode the compressed representation at the bottleneck layer into $\tilde{I}^{M-1}$. This image has the smallest size and contains the low-frequency part; no ground truth is associated smaller than that. The rest D_m denotes single decoding block. Similar interpretation is for E_{M-1} and E_m. The output of a given decoding block D_m can thus be written as $\tilde{I}^m = D_m(D_{m+1}...(\tilde{I}^{M-1}))$. The decoding ability of the AE is enhanced with multi-scale lowpass objective by gradually restoring the high-frequency part of the image into the output (see Fig. 3). Notwithstanding, our ultimate goal is to improve the encoding ability of the AE to remove high-frequency adversarial/noisy perturbations. Below we introduce our network reformation inspired by Laplacian pyramid.

A Laplace pyramid represents an image by the decomposition $LP(I) = [P^0, ..., P^m, ..., I^{M-1}]$, where $P^0, ..., P^m$ are the detail layers (high-frequency bands) of the original image and I^{M-1} the lowest resolution of the image (see Fig. 2 second and fourth rows). The high frequency bands $P^0, ..., P^m$ are disjointed from the lowpass images $I^0, ..., I^m$ in the following way:

$$P^m = I^m - U(I^{m+1}) \tag{1}$$

where U is an upsampling operator.

Applying this pyramid to the decoder, each decoding block D_m is reformed into two modules U and H as shown in Fig. 4: its U module is an upsampling block operates on the output of the previous decoding block D_{m+1} chaining till I^{M-1} of the lowest resolution. U in D_m reverses the corresponding E_m and replaces its last convolutional layer with a deconvolutional layer of stride 2 to upsample the image. The H module is a 2-layer residual block [8] connecting to the mirrored encoding block E_m for high-frequency communication. Assembling them we reconstruct the m^{th} level lowpass image with higher resolution than the previous level (m+1). Analog to (1), the output $\tilde{I}^m$ of D_m is obtained by,

$$\tilde{I}^m = U(\tilde{I}^{m+1}) + H(E_m). \tag{2}$$

Different levels of details are manipulated in the encoding blocks through H: the high-frequency part of the image are gradually compressed from the frontend to the bottleneck of the network.

The overall architecture is illustrated in Fig. 4. Similar to the hourglass structure in Fig. 3, we also adopt a stack of multiple decoding blocks denoted by D_{M-1} to upsample the compressed representation from the bottleneck layer into the size of the $(M-1)^{\text{th}}$ scale lowpass image and associate the first ground truth I^{M-1}. These decoding blocks share the same filter size, stride and number of channels with that of E_{M-1}. Unlike in Fig. 3, the output channel of the last deconvolutional layer of D_{M-1} in Fig. 4 is reduced to 3 (or 1 for grey image) to directly produce the image I^{M-1}. Notice that it is important for D_{M-1} to upsample the latent feature map into a reasonable size (e.g. 7×7) before reducing its channels. The rest decoding blocks D_m consist of U and H modules. Each U module is an upsampling block, which is indeed similar to the naive decoding block in Fig. 3, but with its output channel being 3 (or 1). Each H module is a 2-layer residual block consisting of two 3×3 convolutional layers and one skip connection: the number of channels of the first convoltuional layer is the same with its connected encoding layer, while the number of output channels of the second is set to 3 (or 1) for the same reason above; 1×1 convolutional layer is added on the skip connection to match the number of channels from input to output.

3.4 Loss function

For image classification, we do not alter the training objective: the commonly used loss function is the cross-entropy loss denoted by $\mathcal{L}_{cls}$. For image reconstruction, we adopt the pixel-wise MSE loss between network prediction and ground truth. Referring to Sec. 3.3, the reconstruction loss is defined over the multi-scale output of the AE:

$$\begin{aligned} \mathcal{L}_{rect} &= \mathcal{L}^0_{rect} + \lambda \sum_{m=1}^{M-1} \mathcal{L}^m_{rect} \\ &= \|\tilde{I}^0 - I^0\|_2^2 + \lambda \sum_{m=1}^{M-1} \|\tilde{I}^m - I^m\|_2^2 \end{aligned} \tag{3}$$

where $\tilde{I}^0$ and $\tilde{I}^m$ are the multi-scale reconstruction from the AE (see Fig. 4). Parameter λ is the loss weight between the original scale (I^0) and other scales (I^m). M is the total number of scales which should not be too large as the lowest resolution would become too small to be sensible. In practice we make sure that no loss function is associated for output size smaller than 7×7.

A multi-task loss function is defined over the image classification and reconstruction branches, as $\mathcal{L} = \mathcal{L}_{cls} + \mathcal{L}_{rect}$. Notice that the AE and classification net do not necessarily share the weights of all the encoding layers, we will provide ablation study for this.

The network is in general trained in a *joint learning* manner which is fast and effective. Nevertheless, on natural image dataset, joint learning from scratch can be hard particularly for the AE. In this context, we recommend an *alternative learning* manner: the AE is trained on some epochs alone. Then, the weights of its encoder are copied into the encoding blocks of the classifier, which in turn is fine-tuned for some epochs. Then, the weights of its encoding blocks are copied back to the encoder of the AE. After a few cycles, both the AE and the classifier are on the right track, we can switch to the joint learning. Inference of the classifier is disjointed from the AE (upper branch in Fig. 4).

4 EXPERIMENTS

4.1 Experimental Setup

Datasets. Experiments are conducted on three popular benchmarks for adversarial learning: MNIST, CIFAR-10, and ImageNet. MNIST consists of 70,000 grayscale images of hand written digits, in which 60,000 of them are used for training and the remaining is for testing. CIFAR-10 consists of 60,000 32x32 colour images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images. As for ImageNet, we use the same 1000 test images from the NIPS 2017 adversarial defense challenge [15].

Implementation details. We choose by default ResNet-18 with SGD optimiser trained on classification task. Its four encoding blocks are taken as the encoder for the AE in Defend++. The decoder design refers to Sec. 3.3. For MNIST, the initial learning rate is set to 0.01 and is decayed by a factor of 10 every 100 epochs until 250 epochs. The momentum is 0.9, weight decay is 0.0005, batch size is 256. For CIFAR-10 and ImageNet, the initial learning rate is set to 0.1 and is decayed by a factor of 10 every 100 epochs until 300

Defend++	FGSM	BIM	C&W	DDN	Clean
1 shared block	84.78	53.09	98.95	98.97	99.42
2 shared blocks	85.20	50.52	98.99	98.95	99.39
3 shared blocks	86.48	54.56	**99.14**	**99.16**	99.42
4 shared blocks	**86.81**	**55.37**	99.08	99.10	**99.47**
ours w/o MS-LP	85.42	53.48	99.01	99.05	99.41
ours w/ MS-LI	85.17	46.90	99.03	99.04	99.39
ours	**86.81**	**55.37**	**99.08**	**99.10**	**99.47**
Hourglass	85.39	53.91	99.03	99.05	99.51
Hourglass+SC	84.40	51.61	90.00	99.10	99.42
Hourglass+SA	85.19	54.32	99.05	99.12	99.45

Table 1: Ablation study of different components of Defend++ on MNIST dataset. Classification accuracy on adversarial and clean examples are reported against several adversarial attacks. *ours* is equivalent to *4 share blocks*. *MS-LP* (multi-scale lowpass objective) is applied to all the hourglass structures.

epochs. The momentum is 0.9, weight decay is 0.0005, and batch size is 256. Parameter λ in (3) is set to 0.01.

Adversarial examples. Adversarial examples are generated from the given classification network via representative methods i.e. FGSM [28], BIM [13], C&W [4] and DDN [25]. For FGSM we set by default the magnitude ϵ of adversarial perturbations as 0.2, 0.05 and 0.01 for MNIST, CIFAR-10, and ImageNet, respectively; we also evaluate the performance of different ϵ in Sec. 4.5. BIM and C&W are iterative methods, we set the maximal iteration for BIM and C&W to 100 and 5×20, respectively. DDN is a SOTA attack, we use its public code to generate adversarial examples.

Adversarial defense and baseline. Adversarial training is a straightforward yet effective defense method. We adopt two schemes: one is adversarial training with FGSM examples [28] (denoted by Adv. FGSM) while the other is the proposed adversarial re-training with DDN [25] (denoted by Adv. DDN). We also implement ComDefend [10] using its published code. Our baseline is a classification network without any defense mechanism.

Evaluation protocol. We evaluate the classification accuracy on both the original and adversarial sets.

4.2 MNIST

We provide ablation study regarding the shared layers (ResNet blocks) between the classification net and AE, multi-scale low-pass objective and multi-scale high-frequency communication.

Shared layers between classification and AE. Recalling Sec. 3.4, the AE and classification net does not need to share all encoding layers. Our default classification net (ResNet-18) has four encoding blocks. In Table 1, we ablate the number of their shared blocks from 1 to 4. It turns out sharing four blocks performs best; the classification accuracy on FGSM attack is 86.81 and on DDN is 99.10; the performance of 3 shared blocks performs very close to that of 4 shared blocks. For CIFAR-10 and ImageNet, the best performance occurs when sharing three blocks. If not specified, all experiments below follow this setting.

Multi-scale lowpass objective for AE. Table 1 also presents our method without multi-scale low-pass objective (denoted by ours w/o MS-LP); the result *ours* is indeed the same with that of 4 shared blocks. The proposed MS-LP clearly improves the defense accuracy on nearly every attack as well as the clean image set. To further investigate its effectiveness, we offer the result of using linear interpolation instead for multi-scale ground truth (denoted by ours w/ MS-LI). Its performance is in general inferior to ours. For instance, the accuracy is 85.17 v.s. 86.81 on FGSM; 46.90 v.s. 55.37 on BIM.

Multi-scale high-frequency communication for AE. We compare to the basic hourglass structure in Fig. 3. The result in Table 1 shows that Hourglass produces accuracy clearly lower than ours on every entry. As being suggested, the connection between encoder and decoder in the basic hourglass structure is only through the bottleneck, which is weak. In order to enhance this connection, we could add skip connections similar to U-Net [24], where corresponding encoding and decoding blocks (e.g. E^m and D^m) are connected using identity mappings. The feature tensors from two branches can be either concatenated or element-wise added, which we denote as Hourglass-SC and Hourglass-SA, respectively. Results in Table 1 show no significant improvement over the basic Hourglass, and are even worse on some attacks (e.g. 85.19 v.s. 85.39 on FGSM). Hourglass-SA is slightly better than Hourglass-SC yet is still inferior to ours. Our network reformation offers more elaborate frequency-steering for encoding and decoding blocks, compared to the straightforward skip connection on Hourglass. Hence, it ends up with a more robust defense against various adversarial attacks.

Comparison with representative defense methods. Table 2 compares Defend++ to adversarial training with FGSM (Adv. FGSM), adversarial re-training with DDN (Adv. DDN) [25], ComDefend [10] and baseline (No defense). Defend++ significantly improves the performance of the baseline over any adversarial attack. The baseline and Defend++ are both trained with clean images only. Defend++ is inferior to Adv. FGSM on the FGSM attack (86.81 v.s. 99.77) because Adv. FGSM is specifically trained for this attack. However, Defend++ shows a strong generalization ability compared to Adv. FGSM. For instance, it is 55.37 v.s. 9.86 on BIM. BIM is the strongest attack which causes quite visible image distortion (see Fig. 5). Adv.DDN is indeed adversarial retraining where adversarial examples are online updated. This is very slow. More epochs are needed to obtain better performance [25]. In contrast, Defend++ trains with clean images and is much faster.

Performance of ComDefend is also reported under our setting, which performs close to Adv. DDN but in a grey-box scenario: the attacker is oblivious to the reformer (AE) in front of the classification net [10] (see Sec. 2). In a fair comparison with Defend++, where the reformer is known to the attacker in white-box scenario, ComDefend is no better than the baseline (see ComDefend-wb in Table 2). Our work is similar in spirit to ComDefend that we do not use any adversarial examples for training. Nonetheless, ComDefend adds Gaussian noises (GN) to image representations in the reformer to improve the defending performance. By doing the same, we can further obtain our results (see ours + GN): 90.56, 56.44, 98.97, 98.98 on FGSM, BIM, C&W, DDN and clean examples, respectively; the accuracy is clearly improved by +3.75% and +1.13% on FGSM and BIM, while slightly declined on C&W, DDN and clean examples.

Before the end, two more points are worth noting: 1) there exists a similar representative attack to BIM, the projected gradient descent (PGD) [20]. They have the same multi-step generation process yet PGD uses uniform random noise as initialization. The testing results against PGD attack (100 iterations) on MNIST are indeed

	FGSM	BIM	C&W	DDN	Clean
No defense	31.94	0.00	0.00	0.00	**99.60**
Adv. FGSM	**99.77**	9.86	99.04	99.05	99.51
Adv. DDN	95.21	19.55	98.66	98.66	98.94
ComDefend	85.69	19.04	98.87	98.82	99.43
ComDefend-wb	19.93	0.00	0.00	0.00	99.43
ours	86.81	55.37	**99.08**	**99.10**	99.47
ours + GN	90.56	**56.44**	98.97	98.98	99.41

Table 2: Classification accuracy on adversarial and clean examples of selected defense methods. Experiment on MNIST dataset.

similar to BIM: 0.0, 8.4, 16.6, 18.7, 50.5 for No defense, Adv. FGSM, Adv. DDN, ComDefend, and our Defend++, respectively; Defend++ is clearly the best. On the other hand, for adversarial training with PGD (Adv. PGD) on MNIST, it is very slow and we got accuracy 85.1, 40.1, 98.4, 98.8, 52.3, 97.9 for FGSM, BIM, CW, DDN, PGD, clean examples, respectively; Defend++ is better than this Adv. PGD. 2) We notice that even those degraded versions of Defend++, e.g. ours w/o MS-LP or Hourglass in Table 2, perform much better than the baseline and competitive to other representative defense methods. This validates our idea in general of reinforcing the DNN bottleneck of a classifier with an AE for adversarial defense, which can significantly improve the defense *universality*, without the need of adversarial training or pre-processing head.

4.3 CIFAR-10

Table 3 presents the results on CIFAR-10. Like in Table 2, Defend++ significantly improves the defense performance over the baseline (No defense) on all the attacks. It is also clearly better than Adv. FGSM/DDN and ComDefend on C&W and DDN attacks while inferior to Adv. FGSM on FGSM attack (64.63 v.s. 74.01) and ComDefend on BIM attack (16.76 v.s. 23.60). It maintains a relatively good accuracy on the clean set (93.22) while the others clearly drop (91.30, 86.94, and 87.58). Our Defend++ uses only clean images for training. Similar to that in Table 2, we also add Gaussian noises (GN) during training as in [10]: the results are further improved especially against simple attacks (FGSM and BIM) but the accuracy on clean images drops by one point (see Table 3, ours + GN).

4.4 ImageNet

Results on ImageNet are reported in Table 4. The No defense baseline demonstrates a total failure on BIM, C&W, and DDN while our Defend++ substantially improves it. Defend++ also yields better result than Adv.FGSM on every attack except FGSM, where it is fairly lower as Adv.FGSM is trained with FGSM examples. Notice that experiments on ImageNet are only trained with a subset (around 10k images) of it for fast implementation so the overall accuracy is not very high. Fig. 5 shows examples of attacked images. One can clearly see that BIM attack is more visible.

4.5 Analysis and Discussion

We offer more analysis and discussion on MNIST and CIFAR-10.

Accuracy against various target distortions. We vary the magnitude ϵ of FGSM perturbations for Defend++ from 0.1 to 1.0. The accuracy on MNIST is illustrated in Fig. 7. By adopting Defend++, the baseline curve is substantially shifted to the right. This clearly shows that our method improves the robustness of the classification network. The attacker has to almost double the distortion budget.

Removal of adversarial perturbations. The whole AE is not part of the classification model, but for illustration purpose, we inspect its capability of removing adversarial noises (FGSM) on

	FGSM	BIM	C&W	DDN	Clean
No defense	38.35	0.00	0.00	0.00	**93.80**
Adv. FGSM	**74.01**	11.68	89.63	89.46	91.30
Adv. DDN	68.88	14.42	86.93	86.90	86.94
ComDefend	58.06	**23.60**	77.41	76.28	87.58
ours	64.63	16.76	89.78	**89.83**	93.22
ours + GN	67.42	18.12	**89.95**	88.76	92.05

Table 3: Classification accuracy on CIFAR-10 dataset.

	FGSM	BIM	C&W	DDN	Clean
No defense	13.85	0.00	0.00	0.00	**71.25**
Adv. FGSM	**58.30**	10.65	43.45	44.85	52.75
ours	56.45	**40.85**	**54.85**	**56.35**	62.45

Table 4: Classification accuracy on ImageNet dataset.

Figure 5: Illustration of adversarial examples (zoom in for details).

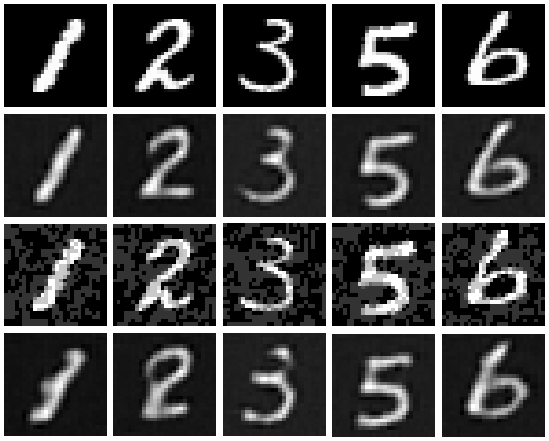

Figure 6: First and second rows: clean images and their reconstructed results via our AE. Third and fourth rows: adversarial images and their reconstructed results via our AE.

MNIST	FGSM	BIM	C&W	DDN	Clean
No defense (VGG16)	39.79	11.68	0.00	17.68	**99.54**
ours (VGG16)	**70.27**	**23.17**	**99.22**	**95.63**	98.84
No defense (ResNet-50)	19.54	0.00	0.00	0.00	**99.65**
ours (ResNet-50)	**87.73**	**54.80**	**99.25**	**99.27**	99.6

Table 5: Experiments on MNIST dataset with classification net being VGG16 and ResNet-50.

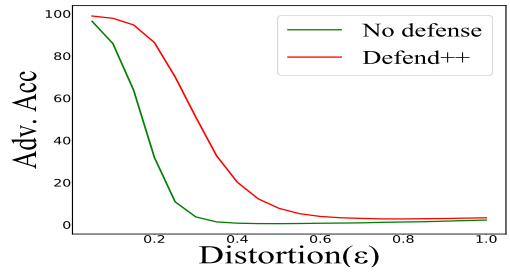

Figure 7: Defend++ against different distortions.

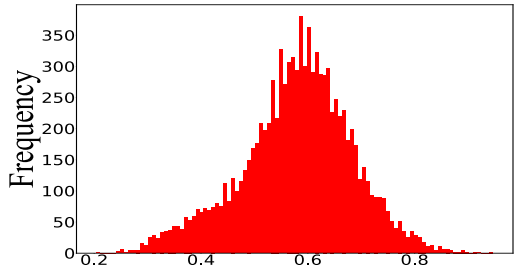

Figure 8: Frequency histogram of R.

CIFAR-10	FGSM	BIM	C&W	DDN	Clean
No defense (VGG16)	35.24	7.23	0.00	0.00	**88.25**
ours (VGG16)	**56.81**	**21.35**	**81.93**	**81.84**	86.60
No defense (ResNet-50)	23.45	0.00	0.00	0.00	**93.97**
ours (ResNet-50)	**79.18**	**48.76**	**90.52**	**90.59**	93.85

Table 6: Experiments on CIFAR-10 dataset with classification net being VGG16 and ResNet-50.

Classifier	VGG16		ResNet-50		ResNet-18	
Attack	VGG16	ResNet-18	ResNet-50	ResNet-18	ResNet-18	VGG16
No defense	39.79	52.73	19.54	33.28	31.94	35.41
ours	**70.27**	**76.09**	**87.73**	**90.47**	**86.81**	**89.39**

Table 7: Evaluation of universality of Defend++ on MNIST.

MNIST. The reconstructed images shown in Fig. 6 are much cleaner than the adversarial images. Quantitatively, we compute the Euclidean distances from the reconstructed image to its corresponding attacked image (D_{ra}) and to its clean image (D_{rc}). Fig. 8 shows that the ratio $R = D_{rc}/D_{ra}$ always falls into the interval $(0, 1)$: The reconstruction is always closer to the clean image than to the attacked one. This proves that the AE succeeds to filter out most of the adversarial perturbation.

Networks and universality. So far, we choose ResNet-18 as our classification net, but Defend++ can adapt any network with the same training procedure (see Sec. 4.1). Here, we select VGG16 and ResNet-50 as the alternative classification nets. For VGG16, we take its 13 convolutional layers as the encoder of AE, every two/three convolutional layers with one pooling layer in the VGGnet are taken as one block. We let AE share the first two block weights with the classification net. For ResNet-50, we take its first four encoding blocks as the encoder of the AE and share their encoding weights. The results are shown in Table 5 and 6 on MNIST and CIFAR-10,

Classifier	VGG16		ResNet-50		ResNet-18	
Attack	VGG16	ResNet-18	ResNet-50	ResNet-18	ResNet-18	VGG16
No defense	35.24	47.49	23.45	41.39	38.35	46.18
ours	**56.81**	**64.22**	**79.18**	**86.13**	**64.63**	**75.45**

Table 8: Evaluation of universality of Defend++ on CIFAR-10.

respectively. Defend++ on VGG16 and ResNet-50 clearly improves the baseline on both adversarial and clean examples, which shows the generalizability of our method on different networks. Notice that 1) the results on ResNet-50 are also slightly better than those on ResNet-18 (Table 2 and 3), as ResNet-50 is a more powerful network; 2) the FGSM, BIM, C&W results (79.18, 48.76, 90.52) on ResNet-50, CIFAR-10 are also comparable to ComDefend [10] under similar setting (e.g. 83, 34, 87 for $L_\infty = 16$ in its Table 5): ours performs better on BIM and C&W attacks. More importantly, ours are pure white-box classifiers, contrary to ComDefend where the upfront reformer is not public (see Sec 4.2).

Defend++ enhances the network defense against universal adversarial perturbations by removing them from the bottleneck representations. From this point of view, we expect that Defend++ can defend adversarial perturbations generated from different models. We use Defend++ trained on one network to defend against adversarial examples generated from another network. Results are reported on FGSM examples amid ResNet-18, ResNet-50 and VGG16 in Table 8 and 7. Having a look at the tables, the first row signifies the classification network we use for testing while the second row signifies the classification network we use for adversary generation. For instance, given a classifier VGG16, its adversarial examples can be generated using the same VGG16 or using ResNet-18. Defend++ is not a *gradient-masking* based approach like [2, 9, 22, 23]; the results show that Defend++ significantly improves the performance over baseline in the transfer setting, which demonstrates its strong universality. Furthermore, comparing the cross-model performance with that using the same model, it is apparent that our Defend++ performs even better on the cross-model. Take the ResNet-50 classifier as an example, its accuracy on attacks from the same ResNet-50 is 87.73 and 79.18 on MNIST and CIFAR-10, respectively; on attacks from ResNet-18 is improved to 90.47 and 86.13 correspondingly. Adversarial examples generated from the same model with the classifier is harder to defend.

5 CONCLUSION

This paper proposes a DNN bottleneck reinforcement scheme for universal adversarial defense (Defend++). It learns an auto-encoder jointly with the classifier by sharing the same encoding parameters of the network. The auto-encoder improves the classifier's compressing ability by removing the adversarial/noisy perturbations at its encoding stage. Multi-scale low-pass objective and multi-scale high-frequency communication is also introduced at training to further improve the robustness of the network. Defend++ is trained with clean images only and without changing the classification structure. Thorough experiments show that, compared to other representative methods, Defend++ is an effective defense against various attacks at very low cost.

ACKNOWLEDGMENTS

This work was supported by the National Natural Science Foundation of China (NSFC) under Grant No. 61828602 and 51475334; as well as National Key Research and Development Program of Science and Technology of China under Grant No. 2018YFB1305304, Shanghai Science and Technology Pilot Project under Grant No. 19511132100, and the French ANR chair SAIDA.

REFERENCES

[1] Yassine Bakhti, Sid Ahmed Fezza, Wassim Hamidouche, and Olivier Déforges. 2019. DDSA: a Defense against Adversarial Attacks using Deep Denoising Sparse Autoencoder. *IEEE Access* 7 (Nov. 2019), 160397–160407. https://doi.org/10.1109/ACCESS.2019.2951526

[2] Jacob Buckman, Aurko Roy, Colin Raffel, and Ian Goodfellow. 2018. Thermometer encoding: One hot way to resist adversarial examples. In *ICLR*.

[3] Nicholas Carlini and David Wagner. 2017. Adversarial Examples Are Not Easily Detected: Bypassing Ten Detection Methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security* (Dallas, Texas, USA) *(AISec '17)*. Association for Computing Machinery, New York, NY, USA, 3–14. https://doi.org/10.1145/3128572.3140444

[4] Nicholas Carlini and David Wagner. 2017. Towards evaluating the robustness of neural networks. In *IEEE Symp. Security and Privacy*.

[5] Gintare Karolina Dziugaite, Zoubin Ghahramani, and Daniel M Roy. 2016. A study of the effect of jpg compression on adversarial images. *arXiv preprint arXiv:1608.00853* (2016).

[6] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. 2015. Explaining and harnessing adversarial examples. In *ICLR*.

[7] Shixiang Gu and Luca Rigazio. 2014. Towards deep neural network architectures robust to adversarial examples. *arXiv preprint arXiv:1412.5068* (2014).

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *CVPR*.

[9] Yingying Hua, Shiming Ge, Xindi Gao, Xin Jin, and Dan Zeng. 2019. Defending Against Adversarial Examples via Soft Decision Trees Embedding. In *ACM MM*.

[10] Xiaojun Jia, Xingxing Wei, Xiaochun Cao, and Hassan Foroosh. 2019. ComDefend: An Efficient Image Compression Model to Defend Adversarial Examples. In *CVPR*.

[11] Linxi Jiang, Xingjun Ma, Shaoxiang Chen, James Bailey, and Yu-Gang Jiang. 2019. Black-box adversarial attacks on video recognition models. In *ACM MM*.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. In *NIPS*.

[13] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. 2017. Adversarial machine learning at scale. In *ICLR*.

[14] X. Li and F. Li. 2017. Adversarial Examples Detection in Deep Networks with Convolutional Filter Statistics. In *2017 IEEE International Conference on Computer Vision (ICCV)*.

[15] Fangzhou Liao, Ming Liang, Yinpeng Dong, Tianyu Pang, Xiaolin Hu, and Jun Zhu. 2018. Defense against adversarial attacks using high-level representation guided denoiser. In *CVPR*.

[16] Zihao Liu, Qi Liu, Tao Liu, Yanzhi Wang, and Wujie Wen. 2019. Feature Distillation: DNN-Oriented JPEG Compression Against Adversarial Examples. In *CVPR*.

[17] Jiajun Lu, Theerasit Issaranon, and David Forsyth. 2017. SafetyNet: Detecting and Rejecting Adversarial Examples Robustly. In *The IEEE International Conference on Computer Vision (ICCV)*.

[18] Chen Ma, Chenxu Zhao, Hailin Shi, Li Chen, Junhai Yong, and Dan Zeng. 2019. MetaAdvDet: Towards Robust Detection of Evolving Adversarial Attacks. In *ACM MM*.

[19] Shiqing Ma, Yingqi Liu, Guanhong Tao, Wen-Chuan Lee, and Xiangyu Zhang. 2019. NIC: Detecting Adversarial Samples with Neural Network Invariant Checking. In *NDSS*.

[20] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. 2017. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083* (2017).

[21] Dongyu Meng and Hao Chen. 2017. Magnet: a two-pronged defense against adversarial examples. In *ACM SIGSAC*.

[22] Aran Nayebi and Surya Ganguli. 2017. Biologically inspired protection of deep networks from adversarial attacks. *arXiv preprint arXiv:1703.09202* (2017).

[23] Nicolas Papernot, Patrick McDaniel, Xi Wu, Somesh Jha, and Ananthram Swami. 2016. Distillation as a defense to adversarial perturbations against deep neural networks. In *IEEE Symp. Security and Privacy*.

[24] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*.

[25] Jérôme Rony, Luiz G Hafemann, Luiz S Oliveira, Ismail Ben Ayed, Robert Sabourin, and Eric Granger. 2019. Decoupling direction and norm for efficient gradient-based l2 adversarial attacks and defenses. In *CVPR*.

[26] Andrew Michael Saxe, Yamini Bansal, Joel Dapello, Madhu Advani, Artemy Kolchinsky, Brendan Daniel Tracey, and David Daniel Cox. 2018. On the Information Bottleneck Theory of Deep Learning. In *International Conference on Learning Representations*. https://openreview.net/forum?id=ry_WPG-A-

[27] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. 2017. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766* (2017).

[28] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. 2014. Intriguing properties of neural networks. In *ICLR*.

[29] Olga Taran, Shideh Rezaeifar, Taras Holotyak, and Slava Voloshynovskiy. 2019. Defending against adversarial attacks by randomized diversification. In *CVPR*.

[30] Weilin Xu, David Evans, and Yanjun Qi. 2017. Feature squeezing: Detecting adversarial examples in deep neural networks. *arXiv preprint arXiv:1704.01155* (2017).

[31] Pu Zhao, Sijia Liu, Yanzhi Wang, and Xue Lin. 2018. An admm-based universal framework for adversarial attacks on deep neural networks. In *ACM MM*.